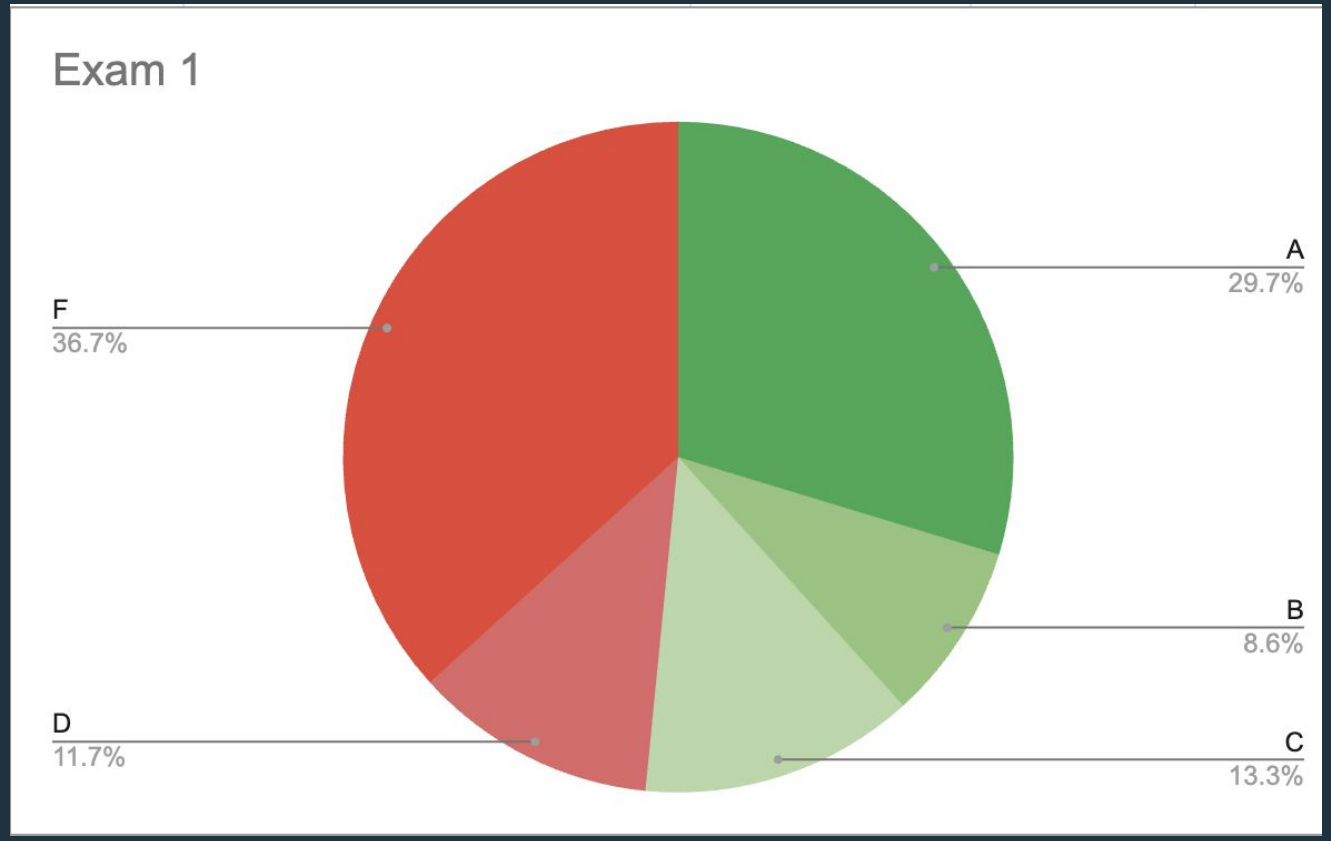
# Exam Review #1

```
<h1>
    CIIC 3015: Intro to
    Programming
</h1>
<date> Oct 9, 2025 </date>
```

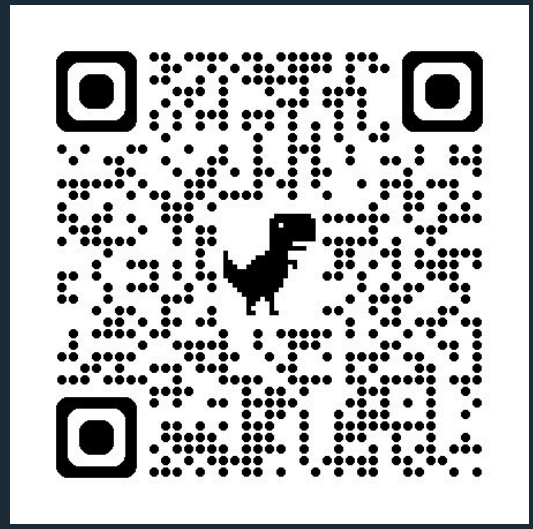
# Exam Stats (avg: 67.15)



# GitHub

Feel free to follow the exercises
with us with the following repo:

https://github.com/UPRM-CIIC3015-F25/ciic3015_postExamReview_1



# PROBLEM 1

 Kinetic Energy 

Write a function kinetic_energy(m,s) that takes an object’s mass m (float,kg) and speed s (float, m/s) and returns its kinetic energy (float) using KE=1/2*m*v^2 rounded to 2 decimals.

Parameters
    m: object mass in kilograms
    s: object speed in meters per second

Returns:
    Object's kinetic energy in Joules (kg·m²/s²).

# PROBLEM 1

 Kinetic Energy 

Write a function **kinetic_energy(m,s)** that takes an object's **mass** m (float,kg) and **speed** s (float, m/s) and **returns** its kinetic energy (float) using **KE=1/2*m*v^2 rounded to 2 decimals.**

Parameters
    m: object mass in kilograms
    s: object speed in meters per second

Returns:
    Object's kinetic energy in Joules (kg·m²/s²).

# PROBLEM 1

```
def kinetic_energy(m,s):
    return round((1/2) * m * s * s, 2)
```

# PROBLEM 1.2

 Cylinder Volume 

Write a function calculate_cylinder_volume(height, diameter) that takes the height (float, meters) and diameter (float, meters) of a cylinder and returns its volume (float) using the formula V = πr²h, rounded to 2 decimals. Note: radius (r) is half of the diameter.

Parameters:

height: Cylinder height in meters (float)

diameter: Cylinder diameter in meters (float)

Returns:

Cylinder volume in cubic meters (float), rounded to 2 decimals.

# PROBLEM 1.2

 Cylinder Volume 

Write a function **calculate_cylinder_volume**(height, diameter) that takes the **height** (float, meters) and **diameter** (float, meters) of a cylinder and returns its volume (float) using the formula **V = πr²h, rounded** to 2 decimals. Note: radius (r) is half of the diameter.

Parameters:

height: Cylinder height in meters (float)

diameter: Cylinder diameter in meters (float)

Returns:

Cylinder volume in cubic meters (float), rounded to 2 decimals.

# PROBLEM 2.1

 Temperature Conversion 

Write a function convert_celsius(c) that takes a temperature c in Celsius (float) and PRINTS the equivalent temperatures in Fahrenheit and Kelvin (both rounded to 2 decimals).
Format: Fahrenheit: X.YY, Kelvin: Z.ZZ.

Formulas:
   Fahrenheit =  (celsius * (9/5)) + 32
   Kelvin = Celsius + 273.15

Parameters
   c: temperature (float) in Celsius

# PROBLEM 2.1

 Temperature Conversion 

Write a function **convert_celsius**(c) that takes a temperature c in Celsius (float) and PRINTS the equivalent temperatures in Fahrenheit and Kelvin (both rounded to 2 decimals).
Format: Fahrenheit: X.YY, Kelvin: Z.ZZ.

Formulas:
   Fahrenheit =  (celsius * (9/5)) + 32
   Kelvin = Celsius + 273.15

Parameters
   c: temperature (float) in Celsius

# PROBLEM 2

 Seconds → H:M:S 

Write a function convert_seconds(total_seconds) that takes a total time in seconds (int) and prints it in the format H hours, M minutes, S seconds.

Parameters:
  total_seconds: Total time in seconds (int)
Prints:
  The time in hours, minutes, and seconds.

# PROBLEM 2

 Seconds → H:M:S 

Write a function **convert_seconds(total_seconds)** that takes a **total time in seconds (int)** and **prints it in the format H hours, M minutes, S seconds.**

Parameters:
  total_seconds: Total time in seconds (int)
Prints:
  The time in hours, minutes, and seconds.

# PROBLEM 2

```
def convert_seconds(total_seconds):
    print(f"{total_seconds // 3600} hours,"
          f" {(total_seconds % 3600) // 60} minutes,"
          f" {total_seconds % 60} seconds")
```

# PROBLEM 3

 Academic Good Standing 

Write a function **good_standing(gpa, credits)** that takes a student's GPA (float) and the number of credits completed (int)

Return True if the GPA is greater or equal to 3.5 and the number of credits is greater or equal to 30.

Otherwise returns False.

# PROBLEM 3

 Academic Good Standing 

Write a function **good_standing(gpa, credits)** that takes a **student's GPA (float)** and the number of **credits** completed **(int)**

**Return True if** the **GPA** is **greater or equal** to **3.5 and** the number of **credits** is **greater or equal** to **30**.

Otherwise returns **False**.

# PROBLEM 3

```
def good_standing(gpa, credits):
    return gpa >= 3.5 and credits >= 30
```

# PROBLEM 3.2

 Check Divisibility 

Write a function check_divisibility( n ) that takes an integer and returns True if it is divisible by 3 and 5 but not by 7. Otherwise returns False.

Parameters:
n: Integer to check divisibility (int)

Returns:
True if n is divisible by 3 and 5 but not by 7.
False otherwise.

# PROBLEM 4.1

 Auto Mileage 

Write a function auto_mileage(miles, fuel) that takes the distance travelled (miles, float) and fuel consumed (gallons, float) by a car in a trip and returns a string that summarizes the car mileage as follows:

- Mileage at or above 40 mpg: Returns "excellent"
- Mileage at or above 30 and below 40 mpg: Returns "good"
- Mileage at or above 20 and below 30 mpg: Returns "average"
- Mileage below 20 mpg: Returns "poor"

Returns:
  String summarizing car mileage (see above)
Hint: mpg is calculated by dividing miles by fuel

# PROBLEM 4

 Triangle Type 

Write a function triangle_type(a, b, c) that takes the three valid sides of a triangle and returns "equilateral", "isosceles", or "scalene" depending on the triangle type.

Parameters: a: side, b: side, c: side

Returns:
- "equilateral" if all three sides are equal
- "isosceles" if two sides are equal
- "scalene" if all three sides are different

# PROBLEM 4

 Triangle Type 

Write a function **triangle_type(a, b, c)** that **takes the three** valid sides of a triangle and **returns "equilateral", "isosceles", or "scalene"** depending on the triangle type.

Parameters: a: side, b: side, c: side

Returns:

- "equilateral" if all **three** sides are **equal**
- "isosceles" if **two** sides are **equal**
- "scalene" if all **three** sides are **different**

# PROBLEM 4

```
def triangle_type(a, b, c):
    if a == b == c:
        return "equilateral"
    elif a == b or b == c or a == c:
        return "isosceles"
    return "scalene"
```

# PROBLEM 5

 Paint Coverage 

Write a function paint_cans(area, sqr_meters_per_can) that takes the area of a wall (float, m²) and the square meters (float, m²) that a single can of paint can cover and returns the minimal number of paint cans required to paint the entire wall.

Parameters:
    area: area of a wall in square meters (float)
    sqr_meters_per_can: square meters(float) that a can of paint can cover

Returns: Number of cans (int) required to cover the wall

# PROBLEM 5

 Paint Coverage 

Write a function **paint_cans**(area, sqr_meters_per_can) that takes the **area** of a wall (float, m²) and the **square meters** (float, m²) that a single can of paint can cover and **returns** the **minimal number** of paint **cans** required **to paint** the entire wall.

Parameters:
    area: area of a wall in square meters (float)
    sqr_meters_per_can: square meters(float) that a can of paint can cover

Returns: Number of cans (int) required to cover the wall

# PROBLEM 5

```
def paint_cans(area, sqr_meters_per_can):
    import math
    return math.ceil(area / sqr_meters_per_can)
```

# PROBLEM 5.2

 Box Fitting 

Write a function large_boxes(small_boxes, sm_boxes_per_lg_box) that takes a total numbers of small boxes (int) and the max number of boxes that fit in one large box and returns the minimal number of large boxes required to accommodate all the small boxes.

Parameters:

small_boxes: numbers of small boxes

sm_boxes_per_lg_box: max numbers of small boxes that fit on one large box

Returns:

The minimal number (int) of boxes required to accomodate all the small boxes

# PROBLEM 6.1

 Movie Night 

Write a function movie_cost(adults, children) that takes the number of adults (int) and children (int) and calculate the total cost of all tickets. Assume that adult tickets cost $10 and child tickets cost $6.50. If the total number of tickets exceeds 4, apply a 15% discount. Value returned must be rounded to 2 decimal digits.

Parameters:
  adults: number of adults requiring tickets
  children: number of children requiring tickets

Returns:
  Total cost (in dollars and cents) of all tickets

# PROBLEM 6

 Tuiting Adjustment 

Write a function calculate_tuition(residency, gpa, credits) that takes student residency ("in" or "out"), GPA (float), and credits (int).
Tuition per credit: $220 for "in", $480 for "out". If GPA ≥ 3.7 and credits ≥ 12, apply 10% discount. Return final tuition rounded to 2 decimals.

Parameters:
   residency: "in" for in-state, "out" for out-of-state (string)
   gpa: Student GPA (float)
   credits: Number of credits (int)

Returns:
   Final tuition cost (float), rounded to 2 decimals

# PROBLEM 6

 Tuiting Adjustment 

Write a function **calculate_tuition**(residency, gpa, credits) that takes student **residency** ("in" or "out"), **GPA** (float), and **credits** (int).
Tuition per credit: $220 for "in", $480 for "out". If GPA ≥ 3.7 and credits ≥ 12, apply 10% discount. **Return** final **tuition rounded to 2 decimals**.

```
Parameters:
   residency: "in" for in-state, "out" for out-of-state (string)
   gpa: Student GPA (float)
   credits: Number of credits (int)
Returns:
   Final tuition cost (float), rounded to 2 decimals
```

# PROBLEM 6

```
def calculate_tuition(residency, gpa, credits):
    if residency == 'in':
        tuition_per_credit = 220
    elif residency == 'out':
        tuition_per_credit = 480

    total_tuition = tuition_per_credit * credits

    if gpa >= 3.7 and credits >= 12:
        total_tuition *= 0.9  # Apply 10% discount

    return round(total_tuition, 2)
```

# PROBLEM 7

 Letter Exam Grade 

Write a function exam_grade(score, course_type) that takes two parameters: a student's score (float, 0–100) and the course type (string, either "regular" or "honors").

For a regular course:
- ≥ 90 → "A"
- ≥ 80 → "B"
- ≥ 70 → "C"
- ≥ 60 → "D"
- otherwise → "F"

For an honors course:
- ≥ 95 → "A"
- ≥ 85 → "B"
- ≥ 75 → "C"
- ≥ 65 → "D"
- otherwise → "F"

Return the letter grade as a string.

# PROBLEM 7

 Letter Exam Grade 

Write a function **exam_grade**(score, course_type) that takes two parameters: a student's **score** (float, 0–100) and the **course type** (string, either "regular" or "honors").

For a regular course:
- ≥ 90 → "A"
- ≥ 80 → "B"
- ≥ 70 → "C"
- ≥ 60 → "D"
- otherwise → "F"

For an honors course:
- ≥ 95 → "A"
- ≥ 85 → "B"
- ≥ 75 → "C"
- ≥ 65 → "D"
- otherwise → "F"

**Return** the letter **grade** as a string.

# PROBLEM 7

```
def exam_grade(score, course_type):
    if course_type == "regular":
        if score >= 90:
            return "A"
        elif score >= 80:
            return "B"
        elif score >= 70:
            return "C"
        elif score >= 60:
            return "D"
        else:
            return "F"
    elif course_type == "honors":
        if score >= 95:
            return "A"
        elif score >= 85:
            return "B"
        elif score >= 75:
            return "C"
        elif score >= 65:
            return "D"
        else:
            return "F"
    else:
        return "Invalid course type"
```

# PROBLEM 7.2

 Shipping Cost Calculator 

Write a function shipping_cost(weight, shipping_type) that takes two parameters: a package's weight in kilograms (float, 0–50) and the shipping type (string, either "standard" or "express").

For standard shipping:

- ≤ 2 kg → 5
- > 2 and ≤ 10 kg → 10
- > 10 and ≤ 20 kg → 20
- > 20 kg → 30

For express shipping:

- ≤ 2 kg → 20
- > 2 and ≤ 10 kg → 25
- > 10 and ≤ 20 kg → 35
- > 20 kg → 45

Return the shipping cost as an integer.

# PROBLEM 8.1

 Baggage Fees 

Write two functions to calculate the total cost of a checked bag with a base fee and overweight surcharges:

1. overweight_fee(weight):
   Takes the total weight of a bag and applies the following rules:
   - If weight < 40 kg → fee = 0
   - If 40 ≤ weight ≤ 50 kg → fee = 10 dollars * (weight - 40)
   - If weight > 50 kg → fee = 20 dollars * (weight - 50)

2. total_bag_fee(weight):
   Calls overweight_fee(weight) to calculate any extra charge.
   The total fee is the base fee of 35 dollars plus the overweight fee.

# PROBLEM 8

 Movie Ticket Cost 

Write two functions to calculate the total cost of a movie ticket with a base price and age-based discounts:

1. age_discount(age):
   Takes the age of a customer and applies the following rules:
      - If age < 5 → discount = full base price
      - If 5 ≤ age ≤ 12 → discount = 50% of base price
      - If 13 ≤ age ≤ 64 → discount = 0
      - If age ≥ 65 → discount = 30% of base price
2. total_ticket_price(age):
   Calls age_discount(age) to calculate any discount.
   The total price is the base price of 12 dollars minus the discount.

# PROBLEM 8

 Movie Ticket Cost 

Write two functions to calculate the total cost of a movie ticket with a base price and age-based discounts:

1. **age_discount(age)**:
   Takes the age of a customer and applies the following rules:
   - **If age < 5 → discount = full base price**
   - **If 5 ≤ age ≤ 12 → discount = 50% of base price**
   - **If 13 ≤ age ≤ 64 → discount = 0**
   - **If age ≥ 65 → discount = 30% of base price**
2. **total_ticket_price(age)**:
   Calls **age_discount(age)** to calculate any discount.
   The total price is the base price of **12** dollars **minus** the **discount**.

# PROBLEM 8

```
def age_discount(age):
    return 0 if age <= 64 else 0.3 if age > 64 else 0.5 if age <=
12 else 1

def total_ticket_price(age):
    return 12.0 * age_discount(age)
```

# PROBLEM 9

 Bonus: Data Bill 

Write a function data_bill(data, customer_type) that takes the data used in a month (float, GB) and a customer type (string, "individual" or "business") and computes the total amount (float rounded to 2 digits) to be billed based on the following criteria:

- Base fee of $25 includes first 2GB
- Plus $8 per each additional GB or fraction
- If consumption above 10GB, add $15 surcharge
- Business customers pay 15% of the total bill

# PROBLEM 9

 Bonus: Data Bill 

Write a function **data_bill**(data, customer_type) that takes the **data** used in a month (float, GB) and a customer **type** (string, "**individual**" or "**business**") and **computes the total amount** (float rounded to 2 digits) to be billed based on the following criteria:

- **Base fee of $25 includes first 2GB**
- **Plus $8 per each additional GB or fraction**
- **If consumption above 10GB, add $15 surcharge**
- **Business customers pay 15% of the total bill**

# PROBLEM 9

```
def data_bill(data, customer_type):
    import math
    bill =  25
    data_left = max(data-2,0)
    bill += math.ceil(data_left) * 8
    if data > 10:
        bill += 15
    if customer_type == 'business':
        bill *= 1.15
    return round(bill,2)
```

# PROBLEM 9.1

 Bonus: Parking Fee 

Write a function parking_fee(minutes) that takes the parking time in minutes (int) and returns the fee (float). Round up to the nearest hour.
Rates:

- First 3 hours = $3.00/hr.
- After 3 hours = $2.00/hr.
- Max daily = $20.

# bProblems

## Practice in Moodle for 5 Bpts

# bProblems

## Practice in Moodle for 5 Bpts

pwd: **bExam1**

# PROBLEM 9: bProb1

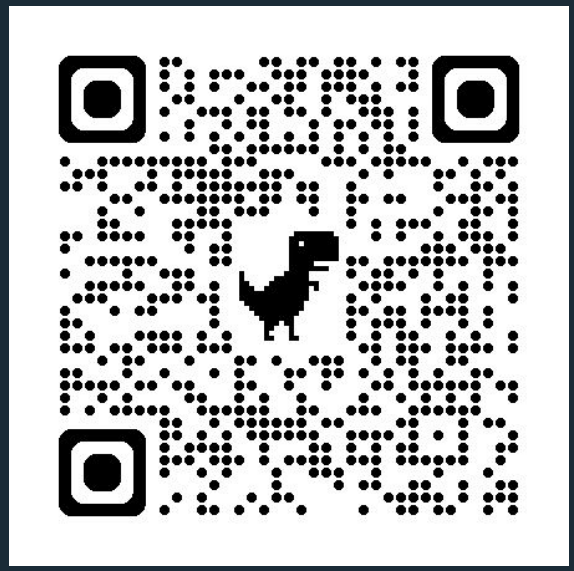
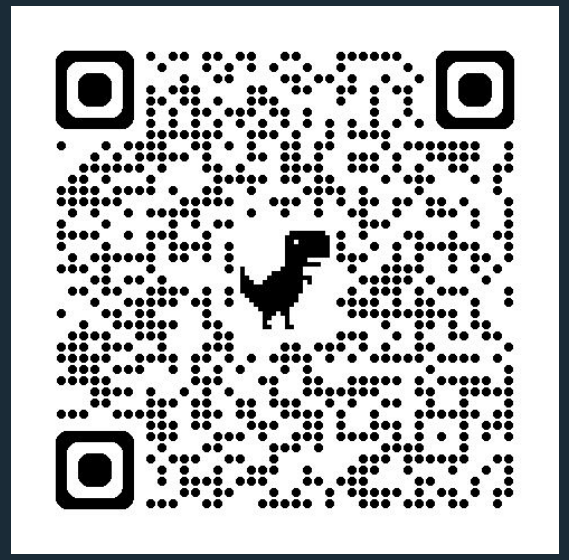
```python
def calculate_tip(bill_amount, tip_percentage):
    tip_amount = bill_amount * (tip_percentage / 100)
    total_amount = bill_amount + tip_amount
    return round(total_amount, 2)
```

# PROBLEM 9: bProb2

```
def calculate_bmi(weight_kg, height_m):
    bmi = weight_kg / (height_m ** 2)
    if bmi < 18.5:
        return "Underweight"
    elif bmi < 25:
        return "Normal weight"
    elif bmi < 30:
        return "Overweight"
    else:
        return "Obesity"
```

# CiR Feedback

https://www.surveymonkey.com/r/cir-fall25-mc



# THANK YOU!